2021. 12. 04. 교육

나정휘
https://justiceHui.github.io/

목차

조합 게임

조합 게임

- 조합 게임 (Combinatorial Game)
  - 두 명의 플레이어 A, B가 참여
  - A, B는 턴을 번갈아 가면서 게임을 함
  - 두 플레이어는 게임의 상황에 대한 모든 정보를 갖고 있음 (포커, 스타크래프트 등은 해당 안 됨)
  - 게임은 항상 유한한 턴 안에 종료됨
    - 일반적으로 마지막에 플레이한 사람이 이김**(Normal Play Rule, 더 이상 행동을 할 수 없는 플레이어 패배)**
    - 마지막에 플레이한 사람이 지는 규칙도 있음(Misère Play Rule)
  - 현재 상태 S에서 A와 B가 취할 수 있는 행동 집합이 F_A, F_B라고 할 때
    - 항상 $F_A = F_B$이면 공정한 게임**(Impartial Game)**
    - 그렇지 않으면 편파적인 게임(Partisan Game, ex. 체스)

  - 오늘은 주로 Normal Play Rule, Impartial Game의 **필승법**을 다룸
  - 최근 2년 연속으로 KOI 1차 대회에 나왔기 때문에, 엄밀한 증명은 모르더라도 내용은 알 필요가 있음

예시) 베스킨라빈스 31

- 두 플레이어는 차례로 턴을 주고 받으면서 1부터 31까지의 수를 차례대로 부름
- 각 플레이어는 한 턴에 1~3개의 수를 부를 수 있음
- **마지막 수인 31을 부르는 사람이 이긴다.** (Normal Play Rule)

게임의 상황

- 게임의 현재 상황을 하나의 “위치”로 생각할 수 있다.
    - 예시: x까지 부른 상황을 x라고 표현하자.
  - 게임을 “종료 위치”로 만든 사람이 승리
    - 예시: 31까지 부른 상황은 “종료 위치”, “시작 위치”는 0까지 부른 상황
  - 게임의 상황을 “종료 위치”로 만들 수 있다면
    - “종료 위치”로 만들면 승리
    - “이기는 위치”라고 하자.
    - 예시: 28, 29, 30은 각각 3, 2, 1개의 수를 불러서 “종료 위치”를 만들 수 있음
  - 현재 상황에서 만들 수 있는 상황이 “이기는 위치” 밖에 없다면?
    - 어떤 행동을 하더라도 상대방이 이기는 위치를 잡게 됨
    - “지는 위치”라고 하자.
    - 예시: 27에서 1, 2, 3개의 수를 불러서 만들 수 있는 상황은 모두 “이기는 위치”이므로 27을 부르는 플레이어는 패배함
  - 현재 상황에서 만들 수 있는 상황 중 “지는 위치” 가 있다면?
    - 상대방을 “지는 위치”로 밀어 넣을 수 있음
    - 현재 상황은 “이기는 위치”
    - 예시: 26에서 1개의 수를 부르면 상대방을 27(지는 위치)로 보낼 수 있음

예시) 베스킨라빈스 31

- 31 : 종료 위치
- 28 ~ 30 : 이기는 위치
- 27 : 지는 위치
- 24 ~ 26 : 이기는 위치
- 23 : 지는 위치
- 20 ~ 22 : 이기는 위치
- 19 : 지는 위치
- …
- 3 : 지는 위치
- 0, 1, 2 : 이기는 위치
- 시작 위치가 이기는 위치이므로 먼저 시작하는 사람이 이김

게임의 상황

- P-위치
  - Previous Player가 이기는 위치
  - 방금 턴을 가진 사람이 이기는 위치
  - 이 상황으로 만든 사람이 이기는 위치
  - Normal Game Rule에서 종료 위치는 P-위치
  - P-위치로 만들 수 있는 수가 존재하지 않는다면 P-위치
- N-위치
  - Next Player가 이기는 위치
  - 턴을 가질 사람이 기니는 위치
  - 이 상황에서 행동을 해야 하는 사람이 이기는 위치
  - 시작 위치가 N-위치면 선공 승리
  - P-위치로 만들 수 있는 수가 존재한다면 N-위치

- 앞에서 봤던 예시에서 3, 7, … , 19, 23, 27, 31은 P-위치
  - 상대방을 3으로 밀어 넣었으므로 Previous Player 승리
- 28, 29, 30은 N-위치
  - 이제 턴을 갖는 플레이어가 각각 3, 2, 1개 부르면 이기므로 Next Player 승리

질문?

동적 계획법의 활용

동적 계획법의 활용

- 게임은 항상 유한한 턴 안에 종료됨
  - 사이클이 없음
  - 게임들의 상태를 DAG로 표현할 수 있다!
  - DP?

예제) BOJ 9655 돌 게임

- N개의 돌이 있다.
- 각 플레이어는 한 턴에 1개 또는 3개의 돌을 가져갈 수 있다.
- 마지막에 돌을 가져가는 사람이 승리

예제) BOJ 9655 돌 게임

- 0 : P-위치 (종료 위치)
- 1, 3 : N-위치 (0으로 이동 가능)
- 2 : P-위치 (1로만 이동 가능)
- 5 : N-위치 (2로 이동 가능)
- …

- `D[0] = 0; D[1] = 1; D[2] = 0;`
- `D[i] = (D[i-1] && D[i-3]) ? 0 : 1;`

- D[N] = 1이면 먼저 턴을 갖는 플레이어 승리

예제) BOJ 11867 박스 나누기 게임

- 두 박스가 있다.
- 한 박스에는 돌이 N개, 다른 박스에는 돌이 M개 있다.
- 각 플레이어는 매 턴마다 아래 내용을 수행한다.
  - 박스 하나를 선택해서 돌을 모두 버림
  - 두 박스가 모두 비지 않도록 다른 박스에 들어있는 돌을 적절히 분배
- 두 박스에 있는 돌의 개수를 모두 1로 만드는 사람이 승리

예제) BOJ 11867 박스 나누기 게임

- (1, 1)은 P-위치
- (N, M)에서 이동할 수 있는 위치는
  - N을 버리는 경우 : (1, M-1), (2, M-2), … , (M-1, 1)
  - M을 버리는 경우 : (1, N-1), (2, N-2), … , (N-1, 1)

- D[N][M] : (N, M)이 N-위치이면 1, P-위치이면 0
- O(NM * (N + M))

```cpp
#include <bits/stdc++.h>
using namespace std;

int D[111][111];

int f(int n, int m){
    if(n == 1 && m == 1) return 0;
    int &res = D[n][m];
    if(res != -1) return res;
    res = 0;
    for(int i=1; i<n; i++) if(f(i, n-i) == 0) res = 1;
    for(int i=1; i<m; i++) if(f(i, m-i) == 0) res = 1;
    return res;
}

int main(){
    int N, M; cin >> N >> M;
    memset(D, -1, sizeof D);
    cout << (f(N, M) ? 'A' : 'B');
}
```

질문?

님 게임

님 게임

- 님 게임 (Nim Game)
  - 여러 개의 돌 더미가 있다.
  - 각 더미는 1개 이상의 돌로 이루어져 있다.
  - 각 플레이어는 비어 있지 않은 더미를 하나 선택해서 1개 이상의 돌을 가져간다.
  - 마지막 돌을 가져가는 사람이 이긴다.

님 게임

- 돌 더미가 5개, 각 더미는 최대 100개의 돌로 구성되어 있다면
  - D[100][100][100][100][100]
  - 돌 더미 N개, 각 더미가 최대 K개의 돌을 갖는다면 K^N
  - 🤔

- 다항 시간 풀이가 있지 않을까?

님 게임

(0,0,0) (0,0,1) (0,0,2) (0,0,3) (0,1,0) (0,1,1) (0,1,2) (0,1,3)
(0,2,0) (0,2,1) (0,2,2) (0,2,3) (0,3,0) (0,3,1) (0,3,2) (0,3,3)

(1,0,0) (1,0,1) (1,0,2) (1,0,3) (1,1,0) (1,1,1) (1,1,2) (1,1,3)
(1,2,0) (1,2,1) (1,2,2) (1,2,3) (1,3,0) (1,3,1) (1,3,2) (1,3,3)

(2,0,0) (2,0,1) (2,0,2) (2,0,3) (2,1,0) (2,1,1) (2,1,2) (2,1,3)
(2,2,0) (2,2,1) (2,2,2) (2,2,3) (2,3,0) (2,3,1) (2,3,2) (2,3,3)

(3,0,0) (3,0,1) (3,0,2) (3,0,3) (3,1,0) (3,1,1) (3,1,2) (3,1,3)
(3,2,0) (3,2,1) (3,2,2) (3,2,3) (3,3,0) (3,3,1) (3,3,2) (3,3,3)

빨간색(P-위치)의 규칙이 보이나요?

님 게임

- 돌 더미들의 돌 개수를 모두 XOR한 값이 0이면 P-위치, 0이 아니면 N-위치
    - 종료 위치는 XOR한 값이 0임
    - XOR한 값이 0이 아닌 상태에서 항상 0이 되도록 바꾸는 방법이 존재
        - XOR한 값의 최상위 비트를 p라고 하면, 더미의 돌 개수 중 비트 p가 켜져 있는 더미가 존재함
        - 그 더미에서 돌을 p개 가져가면 됨
    - XOR한 값이 0이면 다음 상태는 항상 0이 아님
        - 어떤 돌 더미를 건드려서 비트 구성이 달라진다면, XOR 값이 0이 아니게 됨

예제) BOJ 11868 님 게임 2

- 돌 더미의 개수와 각 더미의 돌 개수가 주어진다.
- 선공이 이긴다면 “koosaga”, 후공이 이긴다면 “cubelover”를 출력

```cpp
#include <bits/stdc++.h>
using namespace std;

int main(){
    ios_base::sync_with_stdio(false); cin.tie(nullptr);
    int N, S = 0; cin >> N;
    for(int i=0,t; i<N; i++) cin >> t, S ^= t;
    cout << (S != 0 ? "koosaga" : "cubelover");
}
```

예제) BOJ 16895 님 게임 3

- 돌 더미의 개수와 각 더미의 돌 개수가 주어진다.
- 선공이 이기기 위해서 첫 번째 턴에 취할 수 있는 행동의 개수를 출력

- XOR값이 0이 되도록 만드는 경우의 수를 출력

```
#include <bits/stdc++.h>
using namespace std;

int N, A[1010], S, R;

int main(){
    ios_base::sync_with_stdio(false); cin.tie(nullptr);
    cin >> N;
    for(int i=1; i<=N; i++) cin >> A[i], S ^= A[i];
    for(int i=1; i<=N; i++){
        for(int j=1; j<=A[i]; j++){
            int nxt = S ^ A[i] ^ (A[i] - j);
            R += nxt == 0;
        }
    }
    cout << R;
}
```

질문?

예제) BOJ 11694 님 게임

- 마지막 돌을 가져가는 사람이 패배 : Misère Nim Game
  - 모든 더미에 돌이 1개만 있는 경우 : 더미의 개수가 짝수면 N-위치
  - 1이 짝수 개 있으면 XOR 값은 0

  - 돌이 2개 이상 있는 더미가 존재한다면, XOR 값이 0이면 P-위치, 0이 아니면 N-위치
    - 만약 XOR 값이 0이라면 다음 상태는 0이 아님
    - 만약 XOR 값이 0이 아니라면 다음 상태는 0이거나 1이 홀수 개 있는 상태 (2 1 1 1 -> 1 1 1)

```cpp
#include <bits/stdc++.h>
using namespace std;

int main(){
    ios_base::sync_with_stdio(false); cin.tie(nullptr);
    int N, S = 0, O = 1;
    cin >> N;
    for(int i=0,t; i<N; i++) cin >> t, S ^= t, O &= t == 1;
    if(O) cout << (N % 2 == 0 ? "koosaga" : "cubelover");
    else cout << (S != 0 ? "koosaga" : "cubelover");
}
```

예제) BOJ 18937 왕들의 외나무다리 돌게임

- N개의 외나무다리가 있다.
- i번째 외나무다리는 일렬로 나열된 A_i개의 칸으로 이루어져 있다.
- 모든 외나무다리의 첫 번째 칸에 흰 돌을, 마지막 칸에는 검은 돌을 올려놓은 상태로 시작

- 각 턴마다 자신의 색깔의 돌 중 하나를 이동
  - 같은 다리의 다른 칸으로 움직여야 함
  - 돌을 뛰어넘거나 같은 칸에 머무를 수 없음
  - 두 돌의 거리가 멀어져도 됨
- 돌을 움직일 수 없는 사람이 패배 -> 마지막으로 돌을 움직인 사람이 승리

예제) BOJ 18937 왕들의 외나무다리 돌게임

- 두 돌의 거리가 멀어지는 행동 : 멀어진 만큼 쫓아가면 되므로 생각하지 않아도 됨
- 외나무다리를 하나 골라서 두 돌 사이의 거리를 좁히는 게임
- 돌이 $A_i - 2$개 있는 Nim Game

```
#include <bits/stdc++.h>
using namespace std;

int main(){
    ios_base::sync_with_stdio(false); cin.tie(nullptr);
    string S1 = "Whiteking", S2 = "Blackking";
    int N, S = 0;
    cin >> N;
    for(int i=0,t; i<N; i++) cin >> t, S ^= t - 2;
    string st; cin >> st;
    if(st != S1) swap(S1, S2);
    cout << (S != 0 ? S1 : S2);
}
```

질문?

Sprague-Grundy Theorem

Sprague-Grundy Theorem

- 이걸 유도하고 증명하는 건 너무 어려워서 결론만 나열함
- 증명이 궁금하면 https://tamref.github.io/Grundy1/

Sprague-Grundy Theorem

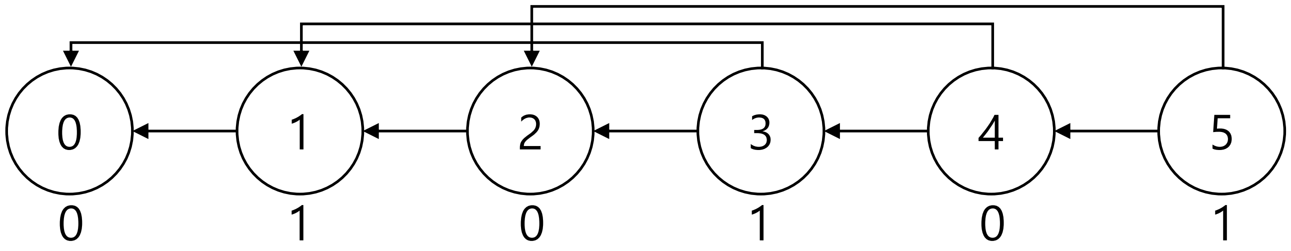
- 앞에서 다룬 게임을 조금 더 이쁘게 정리해보자.
  - 각 상태에서 취할 수 있는 행동(이동할 수 있는 상태)들의 집합을 재귀적으로 표현
    - 아무 상태로도 이동할 수 없는 "종료 상태" : $\emptyset$
    - "종료 상태"로만 갈 수 있는 상태 A의 행동 집합 $A = \{ \emptyset \}$
    - A와 "종료 상태"로 갈 수 있는 상태 B의 행동 집합 $B = \{ \emptyset, A \} = \{ \emptyset, \{ \emptyset \} \}$
    - A로 갈 수 있는 상태 C의 행동 집합 $C = \{ A \}$
  - 더미가 1개인 님 게임
    - 돌이 0개인 경우의 행동 집합 $*0 = \emptyset$
    - 돌이 k(≥ 1)개인 경우의 행동 집합 $*k = *(k-1) \cup \{ *(k-1) \} = \{ *0, *1, \cdots , *(k-1) \}$
      - $\{ *(k-1) \}$: 돌을 하나 가져가면 k-1로 만들 수 있음
      - $*(k-1)$: k-1에서 이동 가능한 모든 상태로 갈 수 있음
  - 더미가 여러 개인 님 게임
    - $*(a_1, a_2, \cdots , a_n) = \cup \{ *(a_1, \cdots , a_{i-1}, x, a_{i+1}, \cdots , a_n) \mid x \in *a_i \}$
    - $*(a_1, a_2, \cdots , a_n)$의 승패 여부는 a_1 xor a_2 xor ⋯ xor a_n이 0인지 판단하면 됨

Sprague-Grundy Theorem

- mex(X) : X에 포함되지 않는 가장 작은 정수 (단, X는 음수가 아닌 정수들의 집합)
  - mex(Ø) = 0, mex{1,2,3} = 0, mex{0,2,3} = 1
- Grundy Number
  - 게임의 상태 S에 대해, G(S) = mex G(s) (단, s는 S의 원소)
    - A = { Ø }, B = { Ø, A }, C = { A }이면
    - G(Ø) = 0, G(A) = 1, G(B) = 2, G(C) = 0
  - S가 "지는 위치"인 것과 G(S) = 0은 동치
    - S = Ø이면 G(S) = 0이고, "지는 위치"라는 것은 자명함
    - 만약 "지는 위치"인 s ∈ S가 존재한다면 G(s) = 0이므로 G(S) ≠ 0이고, S는 "이기는 위치"
    - 만약 S의 원소 중 "지는 위치"가 없다면 G(s) = 0인 s ∈ S가 없으므로 G(S) = 0이고, S는 "지는 위치"

예제) BOJ 9655 돌 게임

- N개의 돌이 있다.
- 각 플레이어는 한 턴에 1개 또는 3개의 돌을 가져갈 수 있다.
- 마지막에 돌을 가져가는 사람이 승리



Sprague-Grundy Theorem

- 여러 게임의 병합
  - 님 게임
    - 더미가 한 개인 님 게임 : $G(*k) = k$
    - 여러 더미가 있는 님 게임 : $G(*(a_1, a_2, \cdots, a_n)) = a_1 \text{ xor } a_2 \text{ xor } \cdots \text{ xor } a_n$? 진짜?
  - 게임1과 게임2를 동시에 진행
    - 두 게임의 상태를 각각 A, B라고 하면 플레이어는 다음 중 한 가지 행동을 취해야 함
      - 게임1에서 한 턴을 수행 (적당한 $a \in A$로 이동)
      - 게임2에서 한 턴을 수행 (적당한 $b \in B$로 이동)
      - 두 게임에서 모두 이동할 수 없는 경우 패배 (A, B 모두 Ø이면 패배 =)
    - 병합된 게임의 상태 : $A + B = \{ (a, B) \mid a \in A \} \cup \{ (A, b) \mid b \in B \}$
    - $G(A + B) = G(A) \text{ xor } G(B)$가 성립함
      - 증명은 생략

예제) BOJ 13034 다각형 게임

- N개의 꼭짓점으로 이루어진 볼록 다각형이 주어진다.
- 각 턴마다 플레이어는 아래 조건을 만족하도록 두 꼭짓점을 고르고 두 꼭짓점을 연결하는 대각선을 긋는다.
  - 이미 그려져 있는 선분과 만나지 않음
  - 변과 일치해도 됨
- 더 이상 선분을 그릴 수 없는 사람이 패배 -> 마지막으로 선분을 그린 사람이 승리

예제) BOJ 13034 다각형 게임

- 변과 일치해도 된다 -> 이각형까지 허용
  - G(0) = G(1) = 0
- n각형에서 대각선을 그리면 i각형과 n-i-2각형으로 나눠진다. (0 ≤ i ≤ n-2)
  - G(n) = mex{ G(i) xor G(n-i-2) }

```cpp
#include <bits/stdc++.h>
using namespace std;

int N, G[1010], C[1010];

int main(){
    ios_base::sync_with_stdio(false); cin.tie(nullptr);
    cin >> N;
    for(int i=2; i<=N; i++){
        memset(C, 0, sizeof C);
        for(int j=0; j<=i-2; j++) C[G[j] ^ G[i-j-2]] = 1;
        for(int j=0; ; j++) if(!C[j]) { G[i] = j; break; }
    }
    cout << (G[N] != 0 ? 1 : 2);
}
```

질문?

예제) BOJ 16877 핌버

- N개의 돌 더미가 주어진다.
- 각 돌 더미에는 P_i(≤ 300만)개의 돌이 있다.
- 플레이어는 자신의 턴마다 더미를 하나 선택해서 임의의 피보나치 수 만큼 돌을 제거한다. (1, 2, 3, 5, 8, 13 등)
- 마지막 돌을 제거하는 사람이 승리

예제) BOJ 16877 핌버

- n번째 피보나치 수는 대략 $O(1.618^n)$
  - 300만보다 작은 피보나치 수는 매우 적음 (31개)

- 각 돌 더미 별로 Grundy Number를 구하면 된다.
  - G(0) = 0
  - G(i) = mex{ G(i – j) } (단, j는 피보나치 수, i – j ≥ 0)

```cpp
#include <bits/stdc++.h>
using namespace std;

int G[3030303], F[31] = {1, 2}, C[32];

int main(){
    ios_base::sync_with_stdio(false); cin.tie(nullptr);
    for(int i=2; i<31; i++) F[i] = F[i-1] + F[i-2];
    for(int i=1; i<3030303; i++){
        memset(C, 0, sizeof C);
        for(int j=0; j<31; j++) if(i-F[j] >= 0) C[G[i-F[j]]] = 1;
        for(int j=0; j<32; j++) if(!C[j]) { G[i] = j; break; }
    }

    int N, S = 0; cin >> N;
    for(int i=0,t; i<N; i++) cin >> t, S ^= G[t];
    cout << (S != 0 ? "koosaga" : "cubelover");
}
```

예제) BOJ 11717 Wall Making Game

- 몇 개의 칸에 X 표시가 있는 H*W 크기 격자가 주어진다.
- 플레이어는 자신의 턴마다 X 표시가 되지 않은 비어 있는 칸을 선택해서
  - 4방향으로 벽을 만나거나 보드 밖으로 나가기 전까지 벽을 설치한다. (X표시가 있는 칸도 벽이 될 수 있다.)
- 마지막으로 벽을 설치한 사람이 승리

예제) BOJ 11717 Wall Making Game

- G[r1][r2][c1][c2] : [r1,r2] × [c1,c2]의 Grundy Number
    - r1 ≤ i ≤ r2, c1 ≤ j ≤ c2인 i j에 대해
    - mex{ G[r1][i-1][c1][j-1] xor G[r1][i-1][j+1][c2] xor G[i+1][r2][c1][j-1] xor G[i+1][r2][j+1][c2] }

- http://boj.kr/d463011821344df3a822c550f19f28a7

질문?